# COMPASS & TAPE


Volume 4 Numbers 3 & 4 Spring 1987

# COMPASS & TAPE

**Volume 4 Number 3 and 4 Spring 1987** Published July 1987

COMPASS & TAPE is the quarterly newsletter of the Survey & Cartography Section of the National Speleological Society, Cave Ave., Huntsville, Alabama 35810, U.S.A. The cost is $4.00 per year for 4 issues. Please make checks payable to Survey & Cartography Section. Include your NSS number for Section membership. Foreign members and subscribers are **welcome!** Rates are US$4.00 per year for surface mail: inquire for air rates. We regret that payment must be in US$, and checks drawn on U.S. banks. The volume runs from the annual NSS Convention: those paying later will receive all back issues. Expiration dates are printed on mailing labels. Volumes 1, 2 and 3 are available for $5.00 each.



***Survey & Cartography Section - 1986/1987***

Chair: John Ganter, address below.
V-C: Dan Crowl, 1170 Torrey Rd, Grosse Pt Woods, MI 48236 313-881-5254
Sec: George Dasher, 109 Shawnee Dr, Buckhannon, WV 26201 304-472-6264

Dues, Subscriptions, Articles,
Maps, Letters, Address Changes,
Comments, Tips, Photos:

**John Ganter**
**Editor, Compass & Tape**
**Department of Geography, 302 Walker Bldg.**
**The Pennsylvania State University**
**University Park, PA 16802**
**H: 814-237-7841 O: 814-865-6421**
**BITNET: GL0 @ PSUVM**

---

**COVER:** Long Caves of California by Bob Richards, 1986.

---

**CONTENTS**



Text Processing Courtesy of the Department of Geography,
College of Earth & Mineral Sciences, The Pennsylvania State University.

# AVERAGING NETWORK ELEMENTS

by David McKenzie
Austin, Texas

## CONTENTS



## APPENDICES



## I. INTRODUCTION

The most extensive cave systems known are networks containing hundreds of passage loops. Cave surveyors consider them the ultimate data-processing challenge when, in fact, the extra information contained in loops should make the surveying task somewhat easier. The difficulties traditionally presented by survey networks relate to two fundamental questions.

First, what is the best way to combine redundant and inconsistent field measurements in order to obtain accurate estimates of location? ("Adjustment" is the surveyor's term for forming final estimates from inconsistent measurement data.) Any procedure should first provide a way to screen out "bad" measurements by somehow comparing them with the entire aggregate of available data. Second, given whatever algorithm we use for obtaining unambiguous results, how can we objectively judge survey accuracy?

In this article I will describe a computer program, NET3, that solves large network problems by means of an unusual least squares algorithm. I say the algorithm is unusual because I have not seen a published description of anything similar, yet for certain applications it is more efficient and direct than standard methods based on Gaussian elimination. NET3, in particular, is compact enough to run on small personal computers and quickly process networks having several hundred nodes.

My goal in writing the program was to eventually include it in a larger package to help cave surveyors organize and present their data. This application is rather specialized, but it provides a clear demonstration of the algorithm while representing an important class of computing problems: connecting simple elements together so that some quantity in the final object is minimized while certain constraints are satisfied exactly. As a simple example, consider the following one-dimensional network problem:

Suppose you wanted to estimate the length of a ladder with 20 irregularly spaced rungs. The only data available are a number of independent estimates of the distances between various pairs of rungs -- for example, between rungs 1 and 5, 1 and 8, 5 and 15, 8 and 15, 15 and 20, 8 and 20, and 5 and 20. In addition, each estimate has an associated error variance, a statistical measure of precision. Since the estimates are independent they are quite likely inconsistent, meaning that lengths "(x to y)" fail to meet the constraints

**(1 to 5) + (5 to 15) = (1 to 8 ) + (8 to 15),**
**(5 to 15) + (15 to 20) = (5 to 20),**
**(8 to 15) + (15 to 20) = (8 to 20).**

The function of NET3 is to take these data and replace them with a new set of consistent estimates, each reflecting the total information available and having an optimally small error variance (which the program also provides). The ladder's length could be derived by adding the 1-to-5 and 5-to-20 results. Or better, we could include a dummy ("infinite variance") 1-to-20 estimate in the original data, enabling us to obtain a variance for our final length estimate.

A more realistic program input might consist

of hundreds of displacement estimates. Also, the network is generally multi-dimensional, with each displacement being a vector with an associated variance-covariance matrix.

In describing the solution method I will try to address two groups of potential readers: surveyors interested in map accuracy, blunder detection, and data processing, and numerically-inclined programmers who might like to experiment with another equation solving method.

## II. THE LEAST SQUARES MODEL

There exists a nice theoretical framework, linear model theory, for solving problems in measurement. Though usually associated with the popular "least squares" method, it is not always a straightforward arithmetic operation. In order to apply it correctly we must first define a set of "observed quantities" (i.e., a set of suitably scaled and transformed measurements) in such a way that they are described reasonably well by a linear statistical model. When least squares is misused (or not used at all), the effect is to throw away potentially useful information, or to overlook spurious data that a comparison with the model might have revealed.

The basic principle, nonetheless, is simple and intuitive: if you were to throw ten darts at a door, another person could estimate from the cluster of darts not only the spot that you were aiming at, but also the consistency of your aim, Such estimates are desired often enough that pocket calculators are commonly equipped with mean and standard deviation keys. Beginning with K. F. Gauss (who in fact originated the least squares method in 1809 to solve a surveying problem), mathematicians have extended the theory of "averaging" to include situations where the elements combined are not just independent samples of one unknown quantity, but instead are correlated observations, each being a distinct function of possibly many unknown quantities.

When the theory is applied rigorously to network problems, it can entail a lot of arithmetic. The well known computer algorithms require workspace and processing time proportional to the square of the number of nodes, or loop junctions, in the network's largest loop system. In the next section I will show how to reduce the constant of proportionality to less than ten percent of its usual value. The numerical method we will use is efficient because it takes special account of the way network observations piece together.

## III. SOLUTION ALGORITHM

The fundamental task to be performed is the least squares estimation of quantities "over-determined" by observations -- a reasonable way to average information while giving more weight to data elements that are more reliable or more precise. We really want more than just an average, however; the network equivalent of "variance" can be used to detect blunders and to judge survey accuracy. From the programmer's standpoint, this means the partial inversion of sparse symmetric matrices. (A matrix is sparse if most of its elements are known to be zero.)

The multi-purpose numerical packages installed at most computer centers can usually solve such problems; however, I know of none that are efficient enough to run on personal computers and still handle networks of significant size. The program described here, NET3, manages to do this by taking an unusually direct route to the solution -- by constructing the partial inverse immediately from network observations, without bothering to set up the original matrix to be inverted. It can, in effect, quickly invert matrices of order several hundred while using only a few thousand bytes of system memory (RAM) for data storage.

More specifically, NET3 obtains network solutions by sequentially modifying the variance-covariance matrix in a least squares linear model. Instead of computing the entire matrix, the algorithm maintains, after each step, only those components necessary for updating selected point displacements and their variances while the remaining observations (vectors) are added to the model. This method of solving least squares problems is not likely to be found in any textbook, although the relevant equations can be derived from a familiar matrix identity which is easy to prove (Ref 2). The equations and certain other details of the NET3 implementation are given in Appendix B.

The algorithm builds up the solution network from component vectors in a step-wise fashion, similar to the way survey measurements are

obtained in the field. At each step the nodes already present in the network belong to two groups: "border" nodes, which connect to vectors still remaining to be added, and the "interior" nodes. The border nodes are represented as row-columns of a variance- covariance matrix, Q, while only those interior nodes for which final displacement/variance information is needed are represented as rows (not columns) of Q. Thus, the numbers of rows and columns may vary either way as the network is generated.

Program execution time is roughly proportional to the number of independent circuits (closures) in the network times the average number of components in matrix Q when a vector between existing nodes is added. Depending on component type, space used is a multiple of the maximum number of rows times the maximum number of columns in Q, the latter being the problem's "bandwidth" (not to be confused with the bandwidth of the corresponding normal equation matrix, which is generally larger). Since the amount of high-speed computer memory available will limit the size of solvable problems, NET3 incorporates a vector sequencing algorithm that tends to produce small bandwidths. (It does this by preventing the unnecessary accumulation of border nodes. See Appendix C.)

Cave surveyors can obtain a good idea of what bandwidth is by imagining having to survey a maze cave while carrying a container of unlabeled tags for use as station markers. It is understood that whenever all vectors to or from any one station have been measured, the station's tag will be returned to the container for later re-use. The bandwidth corresponding to the surveyor's particular plan of attack is simply the total number of tags he requires to complete the survey.

## IV. DESCRIPTION OF NET3

NET3, Version 8-82, was developed for use with Digital Research's CP/M-80 and CP/M-86 operating systems. The source language is Pascal and the compiler is Pascal/MT+, Version 5.5 (CP/M-80 version), also supplied by Digital Research.

The program reads a text file, each line of which defines an observed vector between two named positions (nodes) and an observation error variance. In a survey, for example, the vector could have resulted from actual distance, azimuth, and inclination measurements linking the two nodes or, more typically, from a sequence of component vectors joined head to tail, in which case the variance is the sum of component variances. The vector could, in fact, represent an entire network processed earlier by this program.

A vector may be flagged (with "+") to indicate that no actual observation exists between the named endpoints but that information about their relative location and final variance is wanted. (Assigning a very large error variance would have the same effect.) Conversely, observed vectors flagged with "-" will be omitted from representation as soon as they are used to update the network. This allows networks, which would otherwise be too large for a simultaneous solving for all unknowns, to be correctly processed when only partial information about them is needed.

The program's output is similar to the input, except that each observed vector (not flagged with "-") is replaced by an estimated "network displacement" between the two endpoint nodes -- the result of using the weighted least squares criterion for combining the information contained in all vectors in the file. (An observation "weight" is the inverse of its error variance.) Likewise, each observation error variance is replaced by the "network variance" between the nodes, which is actually the variance of the corresponding displacement estimate assuming the error variances are correct. A motive for using the least squares method of estimation is that the resulting network variances are optimally small given certain reasonable assumptions. (This is the famous Gauss-Markov Theorem. See Ref 4 for details.)

Also provided is the sum of squares of weighted adjustments (residuals) for each vector dimension (SS[i],i=1..Dim), along with the number of independent circuits, NC. A useful indicator of data consistency is the Unit Variance Estimate,

```
UVE = (SS[1]+..+SS[Dim]) / (NC*Dim).
```

If the UVE is much larger than one then the observations do not fit the model, suggesting the presence of blunders or systematic errors. (Most

statistical tests are based on the hypothesis that SS[i] has a Chi-square distribution with NC degrees of freedom.) Otherwise, if it is assumed that the furnished error variances are correct apart from a common scaling factor, the UVE is an unbiased estimate of the variance scaling factor. For a three-dimensional survey, it is convenient to split the UVE into vertical and horizontal parts. Also, see items 4 and 5 in Appendix A, "Notes on Variances".

More specific instructions regarding file formats are given in the NET3 User's Guide (Appendix D).

**Note Concerning Application**

The one-dimensional survey problem is isomorphic to a DC electrical network, where the observed displacements and variances are voltage sources and element resistances, respectively. The resulting network displacements and variances are the potential differences and total network resistances between nodes. The sum of squares is the power dissipated in the circuit. With minor revisions the program can be adapted for the solution/inversion of symmetric matrices. (If negative "variances" are allowed, the matrices need not be positive-definite.)

**Capacity**

The number of represented nodes is currently limited to 255 minus the network dimension, which allows the use of byte-sized pointers in several arrays. The maximum number of vectors will be calculated and should never be a constraint. The compiled code, run time support, and static data will occupy 18-20K bytes of memory, leaving about 36K for dynamic data in a typical 64K CP/M system. Of the latter space we will use

```
(4*(Dim+1)+5)*NV + 4*BW*(BL+Dim) bytes.
```

where Dim is the network dimension, NV is the number of vectors, and BL and BW are the maximum numbers of rows and columns, respectively, in Q. If the algorithm were adapted to use disk storage (by saving border node sub-matrices when Q becomes too large), then the practical minimum RAM requirement (for Q) would be 4*BW*(BW+Dim+1). However, for typical sparse networks, BW is only 5-10% of the number of nodes, which means that networks of more than 200 nodes can easily be solved in a microcomputer's main memory.

There exists a version of the program that stores Q in multiple banks of extended memory starting at a selected 64K bank, in which case Q may contain 65536 real numbers. Execution time is increased by about 5 percent. Use is made of Compupro's 8085/88 dual processor board and a customized CP/M BIOS. In addition, the 1/2-Meg and 1-Meg "semiconductor disks" now on the market are byte-addressable and would require very little software work to interface.

**Accuracy**

In the above formulas "4" refers to the 4-byte, AMD-9511 formatted, floating point numbers, which have about 6.5 decimal digits of precision. When solutions for several survey networks were compared with the corresponding 14-digit results from a Cyber 175/750, the first 6 or 7 digits agreed. The algorithm is unusually stable because the normal equations are not formed and because the elements of Q (covariances) in our case happen to be positive numbers of small range. It also helps that we are usually solving for displacements and variances between adjacent nodes, not values with respect to an arbitrary reference.

**Performance with an Example Problem**

ELLIPSE, a more complete survey data processing program, first organizes sight vectors into strings and strings into independent loop systems before the solution method is finally applied to each system separately. An unusually large loop system for a cave survey is the North Maze of Cueva de Kaua, Yucatan, which I've often used as a benchmark to compare methods. It has 352 strings, 199 junctions (nodes), and 154 loops. By starting at the base survey station and making no attempt to optimally sequence the vectors, I used ELLIPSE to generate an input file for NET3. The bandwidth, without sequencing, was found to be 72, and an older one-dimensional version of the program required 24.5 minutes for a complete solution using extended memory and software arithmetic (16.7 minutes with a 9511 math chip).

By experimenting with the sequencing

algorithm now incorporated in NET3 (see Appendix C for details), it was found that a bandwidth of 11 was possible with any of 27 starting nodes. The numbers of starting nodes that provide each bandwidth are as follows:

| BANDWIDTH: | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FREQUENCY: | 27 | 42 | 14 | 2 | 73 | 9 | 15 | 0 | 10 | 0 | 0 | 7 |

In this example the mean bandwidth is 14.3, with 21 percent of the nodes producing bandwidths greater than 15, the largest being 22. (I also applied the Reverse Cuthill- McKee algorithm to all nodes in this system to find a minimal bandwidth of 22 for the normal equation matrix.)

By choosing starting nodes with bandwidths 11, 15, and 22, vector orderings were produced that led to NET3 solution times (all three dimensions) of 7.2, 7.3, and 8.4 minutes, respectively, using software arithmetic. In the first case, less than 30 percent of the (non-extended) memory available for work area was actually used.

Although sequencing from a specified node in this example takes less than 10 seconds, it is evident that examining all possible starting nodes during each program run would hardly be profitable. The current version of NET3 simply takes the first vector endpoint as a starting node. If the resulting sequence requires too much memory, the remaining nodes are examined in turn.

An important advantage of the algorithm is its ability to efficiently solve for vector subsets, or simply the UVE. In the above example, when all vectors are flagged with "-", the program takes 30 seconds to process the vectors and output a UVE. It would take a little longer to obtain a few solution vectors.

The above times were achieved on a Compupro 8085/88 Dual- processor based system running at 6 Mhz. (The software runs on the 8085.)

## V. FURTHER INFORMATION

This is a review copy of an article to be published in some form or another. I want to thank Pete Lindsley, Robert Thrun, and Martin Heller for their interest and support. If you would like to share results of experiments using different methods, programming languages, or hardware, please contact me at the following address: David McKenzie, 3300 Hemlock, Austin, TX 78722.

[This previously-unpublished manuscript was written in December, 1982. --Ed.]

## VI. REFERENCES

1) Heller, Martin (1980) "Toporobot: Hohlenkartographie Mit Hilfe des Computers," REFLEKTOR No. 2/1980, p.5+. Also personal communmication.

2) McKenzie, D.W. 1977. "The Analysis and Adjustment of Survey Networks". M.A. Thesis, Univ. of Texas at Austin.

3) Mikhail, Edward M., 1976. Observations and Least Squares, IEP, New York. (An introduction to classical survey adjustment using standard numerical methods.)

4) Rao, C. Radhakrishna 1976. Linear Statistical Inference and its Applications, Second Edition. John Wiley and Sons. (Contains a general introduction to Gauss-Markov theory.)

## APPENDIX A

## NOTES ON VARIANCES

Computing the network variances is more difficult than computing the displacements alone -- in the same way that inverting a matrix is harder than "solving with respect to a right-hand side". Surveyors who are uncertain about what the variances are, or who doubt their usefulness, should review the following: (The superscripts "'" and "$^{-1}$" denote the matrix operations of transposition and inversion, respectively.)

1) Consider a set of random vectors -- perhaps a set of line segments "vibrating" in such a way that the length and orientation of any segment at a future time can be characterized by a probability function. Our objective might be to estimate from a sample, or snapshot, the means of those vectors when the means are known to form a connected network. In other words, we know which endpoints attach to which, but we

also observe that the sampled vectors don't quite fit.

If our estimation method is a function of the data alone, then our estimates will also be random vectors with statistical properties. The variance of a random vector is a measure of its expected squared deviation from the mean. Therefore, an additional goal might be to calculate the variances of our estimated means when the variances of the observed vectors have been specified.

If we regard the error in a vector observation as a column vector whose components are random variables with zero means, then the error variance is defined as the expected value (mean) of the error times its transpose. It follows that our variances are symmetric nonnegative definite matrices (or nonnegative numbers if the network is one-dimensional). With the assumption of normal distributions, we can view the elements of this matrix as the coefficients of a quadratic equation describing ellipsoidal surfaces of constant error probability.

2) Suppose, for example, that a three-dimensional random vector were viewed with respect to the coordinate frame that aligns with a particular sample, or observation, of that vector. The random vector could then be considered to have "normal", "transverse", and "vertical" error components that are statistically independent with variances Vn, Vt, and Vv, respectively. This assumption of independence would be realistic if the vector were derived from three spherical-coordinate measurements and the expected errors were reasonably small. One scheme I've used in ELLIPSE, a cave survey data processing program that runs on a large mainframe, is to assign default component variances with the following formulas:

```
Vn = Vp,
Vt = Vp + Va * Lh * Lh,
Vv = Vp + Vi * L * L,
```

where L and Lh are the observed vector's total length and horizontal length, Vp is a target positioning error variance, and Va and Vi are the azimuth and inclination error variances in squared radian units. (Some cave surveyors record the average of a foresight-backsight pair of measurement sets. Besides guarding against blunders, this practice effectively halves the azimuth and inclination error variances.)

If the observed vector were fortuitously aligned with the positive X-axis of the survey frame of reference, the estimated error variance with respect to the X-Y-Z coordinate frame would be the diagonal matrix

```
V = diag(Vn, Vt, Vv).
```

More generally, if the vector were not so favorably aligned, but still had the same error properties, its variance would have to be represented by the non-diagonal matrix

```
V = J * diag(Vn, Vt, Vv) * J',
```

where J is the rotation matrix that transforms coordinates in the observed vector's frame of reference to coordinates in the X-Y-Z frame of reference. This matrix can be computed from the observed vector, X = (x y z)', as

```
          | x/L     -y/Lh    -x*z/(L*Lh) |
          |                              |
J =       | y/L      x/Lh    -y*z/(L*Lh) | .
          |                              |
          | z/L       0          Lh/L    |
```

(Note that J is orthogonal: JJ' = J'J = I. Also, programmers should prepare for the case Lh = 0.)

A cave survey typically consists of many long strings of such vectors. The variance of a string (a vector sum) is obtained by simply summing up the V matrices of the component vectors.

3) A three-dimensional survey consisting of spherical coordinate measurements can be modeled as above. However, it is often more practical to adjust the model slightly to considerably reduce the computation costs, especially when the error variances are rough estimates to begin with. The least serious perturbation is to reshape the error ellipsoids so that their planes of symmetry align with a common coordinate system. The variances then become diagonal matrices and the network effectively breaks into three one-dimensional networks. (Martin Heller takes such an approach in his program, TOPOROBOT (Ref 1). He solves

for the location estimates but not for the network variances.)

A rougher approximation is to assume additionally that each ellipsoid has the same shape (not size), in which case a simultaneous solution for all coordinates is possible while storing each variance as one number. For example, assigning each error component a variance proportional to vector length produces the familiar "compass rule" adjustment. So that it can solve very large networks, this version of NET3 is configured to represent variances as nonnegative real numbers; the benefits described below can still be appreciated in most cases.

4) There are well-established methods for deriving confidence regions for location estimates in linear models (Ref 4). A-priori estimates of precision are based on the network variances alone, which depend on the observation variances and how the network is connected together. How well the observations fit the model is reflected in the residual sums of squares, which can be used to revise the network variances and to produce post-priori confidence regions. (Usually, a variance is revised by simply multiplying by the UVE.)

To continue with the example in note 2, observe that the relation

$$E' V^{-} E = C,$$

where E is an error vector and C is a constant, is the equation of an ellipsoid in the components of E. The constant C = 7.81 gives us a 95-percent confidence region for a 3-dimensional error vector with mean zero and variance V. (The corresponding C for dimensions one and two are 3.84 and 5.99, respectively.)

5) A powerful technique for detecting blunders in observation data depends on knowing how much each observation's separate deletion would reduce the sum of squares (Ref 2). The availability of network variances allow these quantities to be efficiently computed and compared with what would be expected if there were no blunders. Specifically, if an observed vector X with error variance V were deleted from a network with corresponding displacement x and variance v, the total sum of squares, SS=SS[1]+..+SS[Dim], would decrease by

$$Se = (X-x)'(V-v)^{-}(X-x).$$

With the assumption of no blunders, the statistic

$$F = (Se/Dim) / ((SS-Se)/(Dim*(NC-1)))$$

has a "central F-distribution" with parameters Dim and Dim*(NC-1). Thus, observations with unusually large F-statistics (F >> 1) can be considered suspect in a screening procedure. (Note that the denominator in F would be the UVE after deletion.) In addition, the adjustment that would have been applied to the observed vector if it had been assigned infinite variance (zero weight) is easily obtained as

$$Xe = - V(V-v)^{-}(X-x).$$

In surveying, this vector (with sign reversed) will often match a mistake discovered later in transcribed field notes. A simple error analysis program, NETERR, was developed in conjunction with NET3. It compares the input and output files of NET3 to produce, for each vector, the statistics Xe, F, and the UVE after deletion. The current version works only with 3-dimensional data and computes separate horizontal and vertical error statistics (See Appendix E).

6) If A and B are distinct networks of vectors with two nodes in common then the final solution for A, when considered as part of the combined network, could be obtained by representing B as a single vector "observation" whose displacement and error variance had been obtained by first processing B separately to get a network displacement/variance between the two attachment nodes. The result would be virtually the same if all vectors in both networks had been processed simultaneously. If NET3 were revised to read (and store) variance-covariance matrices corresponding to multiple attachment nodes then networks of practically unlimited size could be solved in stages.

## APPENDIX B

## NETWORK MODIFYING EQUATIONS

While generating the solution network, NET3 maintains two internal data structures: the rectangular Q-matrix whose size and logical row/column arrangement dynamically changes, and the static vector array. We denote them as:

```
                --> Border node positions    -->   Q01  Q02 ... Q0b
                |   with respect to root node
                |                                  ------------------
                |                                  Q11  Q12     Q1b
                |                                  Q21  Q22     Q2b
                |   Border node variance-            .          .
                |   covariance matrix        -->     .          .
Q-Matrix:       |    (symmetric)                .          .
                |                                  Qb1  Qb2 ... Qbb
                |
                |                                  ------------------
                |                                  Qb+1,1 .. Qb+1,b
                |   Interior node covariances      Qb+2,1 .. Qb+2,b
                |   with respect to border   -->     .          .
                |                                    .          .
                --->                               Qb+k,1 .. Qb+k,b
```

Vector array:

```
    Displacement/variances between border          (x01, v01)
    and interior nodes which, like Q, are          (x12, v12)
    revised with each addition of a new      -->   (x02, v02)
    vector between existing border nodes            .    .

                                                                .    .
    The next observation to be added               -----------
    (s and/or t is a border node)            -->   (xst, vst)
                                                                (x--, v--)
    Vectors not yet added to the network,          .    .
    some of which are attached to new nodes  -->   .    .
                                                                (x--, v--)
```

If the dimension of the network is m, then Xst and the elements Q0j in the top row of Q are each m-component row vectors, while Vst and the remaining matrix elements, Qij, represent submatrices of order m. In NET3 it is assumed that all observation variances are scalar multiples of the unit matrix, so Vst and Qij are actually stored as scalars (reals).

In addition to the two main structures, the program maintains an m-vector SS, whose components are the residual sums of squares for each network dimension, and an integer NC, the current number of network circuits. The number of degrees of freedom is m * NC. The unit variance estimate (UVE) for dimension p is [SS]p/NC.

When the network is revised with a new observation, (Xst, Vst), the program must perform one of two basic kinds of operations on the data structures. First, a node addition must occur when node s is in the border set and node t is a new (external) node. If node t is to become a border node (meaning that it connects to vectors still remaining to be added) then the program will allocate one row and one column of Q to the new node. The matrix will become larger only if node s must still be maintained as a border node. In any case, if node t is added, the new elements of Q are simply derived as follows:

Node Addition --

```
    Q0t := Q0s + Xst,
    Qit := Qis (for i not 0),
    Qti := Qsi (for i <= b),
    Qtt := Qss + Vst,
    SS, NC and the vector array are unchanged.
```

If s happens to be the root node, then it is not explicitly represented in Q, as the row/column elements are necessarily zero. A complication arises, however, when the root node ceases to be a border node. In order to delete it (or make it an interior node) we must change the network reference. For example, to move the reference to the node currently occupying column s of Q we make the following transformation:

**Change of Reference --**

```
Q0s := -Q0s,  Q0j :=  Q0j - Q0s,
Qss :=  Qss,  Qsj :=  Qss - Qsj,  Qjs :=  Qss - Qjs
Qij :=  Qss + Qij - Qis - Qsj  (for i,j not 0 or
s).
```

Column s will now correspond to the old reference node. Although moving the reference is not strictly required, it reduces the bandwidth by one and helps limit the size range of elements Q0j.

The second basic operation is a circuit addition, that is, attaching a new vector (Xst, Vst) between existing border nodes. This requires that both Q and the array of previously added vectors be revised (Cj' is Cj transposed):

**Circuit Addition --**

```
Qij   :=  Qij - Ci * Mst * Cj',
xij   :=  xij - C0 * Mst * (Cj-Ci)',
vij   :=  vij - (Cj-Ci) * Mst * (Cj-Ci)',
[SS]p :=  [SS]p + [C0]p * [Mst * C0']p,
NC    :=  NC + 1,
```

```
where  Ci = Qit-Qis,  C0 = Q0t-Q0s-Xst, and
                Mst = 1/(Vst+Qss+Qtt-Qst-Qts).
```

Most of the work is in revising Q (first equation). In NET3 this is done with approximately one floating-point multiplication per matrix element. Finally, if either of the two attachment nodes is resolved, it is deleted from Q or else moved to the interior set.

**Additional Notes**

1) The above transformations should not be regarded as explicit program statements. The notation "A := B" means "new version of A takes value of old version of B".

2) Note that Qij' = Qji. To help simplify housekeeping, NET3 stores the complete symmetric submatrix of Q corresponding to the border nodes. (Recall that Qij = Qji = Qji' when Qij is a scalar.) Data elements are never actually moved; we instead revise a structure of pointers associated with Q.

3) The algorithm shares a characteristic with other "sparse" methods in that the housekeeping details of managing workspace can make the program's code seem quite convoluted. Nonbanded versions of the algorithm are easier to implement. I wrote an HP-41CV calculator program which stores the input vectors in packed form on cards or in program memory. It stores the upper half of a symmetric Q-matrix, which is considered the result, in data memory. The user must keep track of which nodes are being deleted or retained. For example, the input vector "3 -5 .4 10" will replace the current "node 5" with a new "node 5" displaced 10 units from "node 3".) The program will compute complete inverses with orders up to 20, or partial inverses with bandwidths up to 20. A network with 87 nodes, 150 loops, and a bandwidth of 7 took about 45 minutes to process.

4) An interesting variation of the algorithm for one-dimensional problems is to maintain a Z-matrix instead of a Q-matrix, where Zij is the current displacement variance between nodes i and j: Zij = Qii+Qjj-Qij-Qji. Similar modifying equations can be derived. This representation for networks is attractive because it does away with the arbitrary reference node.

5) My test problems have shown the algorithm to be quite stable -- more accurate, in fact, than the popular Choleski method applied to the corresponding set of normal equations. However, since zero observation variances are allowed, it is possible to present the algorithm with fundamentally inconsistent data. This is revealed in the attempt to add a vector with zero variance to a network whose current displacement variance is already zero, resulting in Vst+Qss+Qtt-Qst-Qts=0. NET3 will inform the user of this, ignore the vector, and continue processing.

## APPENDIX C

## NET3 SORTING ALGORITHM

NET3 solves a network problem in two stages. The first stage is a sort, or indexing, of the input vectors in order to make the actual solution (second stage) more efficient. The sort is accomplished by adding edges, one by one, to a connected graph, initially consisting of an assumed or specified starting node. A node in the network is considered "resolved" when all edges that connect to it have been added. At each step,

the program examines the remaining edges which attach to unresolved nodes and assigns them a ranking as follows:

A "closure" between existing nodes that resolves:

1) two nodes.
2) one node.
3) zero nodes.

A node addition resolving (including added node):

4) two nodes.
5) one node.
6) zero nodes.

Edges in the last three categories are subranked according to the "score" of the potentially added node. For each unadded edge connecting this node to another existing node, we add "2" to the score if that edge would resolve the existing node, or "1" otherwise. For each edge connecting the potential node to another exterior node, we subtract "1" from the score. Edges in the first three categories are scored the same. We would finally add the edge in the lowest numbered category with the highest score, deciding not quite arbitrarily between ties. Edges connected to the "older" unresolved nodes are favored.

The bandwidth corresponding to any particular starting node is defined as one less than the maximum number of unresolved nodes in the graph during any stage of the process of adding edges. An efficient algorithm for finding edge orderings that give the smallest possible bandwidth probably doesn't exist. Likely, a better heuristic method than the above can be discovered by modifying the score function.

## APPENDIX D.

## NET3 Version 8-82 USER'S GUIDE

The program reads a file of vector data whose name (without the extension) is typed on the CP/M command line following the program name. The extension ".NET" is always assumed for the data file. For example, type "NET3 DAT20" in response to the CP/M command prompt and the program will expect to find the file "DAT20.NET" on the default drive. (A drive can also be specified, as in "B:DAT20".)

**Data File Format**

The first non-blank character on a line can signify one of three types of vectors. A plus sign (+) indicates that no true observation exists between nodes nam1 and nam2, but that a network displacement and variance is wanted. A minus sign (-) means that a final displacement/variance is not needed, but that the observation will be used to revise the network. Any other character will be taken as the start of nam1, in which case an observation is present and a network displacement/variance will be computed. The remaining items on the line are separated by spaces and have the following arrangement:

nam1,nam2 -- Two distinct node names. The program regards the nam2 first five characters of each name as significant.

E -- Error standard deviation (E >= 0). Normally a decimal number, but the program could be revised to accept a variance-covariance matrix when the observation is a vector with correlated components. In this version the components are assumed uncorrelated, each with variance V = E*E (or a constant multiple of V).

X -- Observed displacement vector from nam1 to nam2. NET3 accepts three decimal numbers as coordinates. Trailing blank fields default to zero.

Since error components are assumed uncorrelated, each network dimension is in effect processed independently of the other dimensions. The program can safely be used for one-dimensional and two dimensional problems. For example, in a three-dimensional survey, if the vertical component error variances differ from those of the horizontal components (in ways other than a constant scaling factor) then the vertical and horizontal components can be processed as two separate problems.

The vectors may appear in any order in the data file as long as they collectively represent a connected network. This means that a sequence is possible in which every vector but the first has an endpoint appearing earlier in the sequence (on a vector with finite variance). This condition insures that all network variances are finite.

**Output File Format**

When the program is executed, the current capacities with respect to vectors, nodes, and workspace (in 4-byte units) are written to the terminal screen. The first overlay, NET3.001, reads the vector data into memory and the second overlay, NET3.002, performs a network sort (indexing) starting at the first node. If the resulting bandwidth is too large then indexing is attempted using different starting nodes. When indexing is complete, the bandwidth, bandlength, and numbers of vectors, nodes, and loops are reported. If available memory is adequate, overlay NET3.003 is invoked to solve the network.

Progress during each of the reading, indexing, and solution phases is indicated on the terminal screen by a row of asterisks -- one asterisk for each five vectors processed. Finally, the UVE is displayed, and the user is given the option of creating a text file with extension ".ADJ" containing the results (overlay NET3.004).

The optional output file can be processed, along with the original data, by programs doing error analysis and/or plotting. The first line contains the number of components in each vector, the number of loops (closures) in the network, and the component sums of squares. The remaining lines comprise the solution vectors. The first item on a line is the vector's sequence number in the input file. (Vectors which were flagged with "-" are excluded.) The network variance appears next, followed by the components of the estimated displacement. Six-digit scientific notation is used for the four values.

## APPENDIX E

## NETERR Version 8-82 USER'S GUIDE

A simple network error analysis program, NETERR, was written to make use of the variance and displacement information generated by NET3. An obvious enhancement would be graphic output that would highlight (with more vivid colors, for example) vectors with outstanding error statistics.

The CP/M command is "NETERR filname", where filname.NET and filname.ADJ are both assumed to exist on the default drive. The ADJ file must have resulted from running program NET3 with filname.NET as input.

A network error analysis is performed by computing the effects of independently deleting each observed vector from the statistical model. (See Appendix A for an explanation of the theory.) The user will have the option of displaying the results on the screen or writing them to a file, filname.LST. If you choose to display the results, be prepared to toggle CNTRL-S as necessary. You will get just one pass through the data.

For each solution vector in filname.ADJ (excluding those flagged with "+" in filname.NET), a line will be printed containing two endpoint names, F and UVE for horizontal error, F and UVE for vertical error, and the correction (X,Y,Z) that would need to be applied to the observed vector if it were assumed to be a blunder (given zero weight). The UVE is the network UVE after the vector's deletion. Vectors should be considered suspect if the statistic F is relatively large. At the beginning of the printout are the horizontal and vertical UVE's prior to any deletions.

Although in theory we can devise formal rules for rejecting vectors (balancing probabilities of accepting good observations and rejecting true blunders), the statistics are probably more useful as simple guidelines. With experience, the user should learn to know what to expect, depending on the quality of his surveys and his method of assigning error variances. To start with, I suggest he introduce one or more "blunders" in a typical data file, then run NET3 and NETERR to see what effect they have on the statistics.

# Lighting Up Your Suunto

by Richard Market
Princeton, Indiana

Ever lie in a tight muddy crawl with water dripping on you and try to shine a light over the top of a Suunto in order to read it? How about hanging it out over a nasty lip while trying to hold a light over the Suunto, while still hanging on ?

There is another way. You do not have to hold a light over a Suunto to read it. The dial can be lit very easily using a common red LED (light emitting diode).

All you need is: (1) An LED. (Note: Some LEDs have steel leads which would interfere with the compass reading. Try a magnet before you buy.) (2) Small flexible lead wire, 20 to 22 AWG. The more strands in it the better. (3) Battery: voltage of your choice. (4) Resistor: The value of the resistor is determined by the battery voltage and the amount of current you need for LED operation.

Resistor = Battery Voltage - LED Voltage / Current

The current of the LED should be adjusted to please the eye of the user. You can light the dial up with only 2 MA (milliamps), while some eyes require more light. You can increase the current up to 15 MA on most LEDs which would be too much light for some eyes. The first few trips you may want to use a variable resistor to help you choose.

Depending on how deep you want to get into this, you can use resin or RTV silicone to attach the LED to the Suunto. You can also make an 'L' bracket which mounts under the eyebolt which holds the cord. Either one of the above does not permanently modify the Suunto. However, if you want, you can drill into into the Suunto to attach or implant the LED. Your imagination is the only limiting factor.

The LED should be mounted just a little off center to enable you to read the dial in daylight or with a overhead light if the unit should fail in the cave. Replace the cord which came with the Suunto with 1/8-inch braided cord. Remove and discard the center of the cord, and feed the lead wire through the center of the braid. Now you have a flexible wire running through a braided sheath from the resistor/battery pack to the Suunto, with the sheath holding the Suunto to eliminate any mechanical pull on the wire. You may want to make the cord a little longer so that the battery/resistor pack can be placed in a pocket, on the helmet, or clipped to the collar behind the neck.

You may want to add a switch somewhere in the system, or you could turn the battery around backwards, or disconnect it until ready to use. If you are only using 5 to 10 MA it would take some time to run the battery down. (With a 9 volt Nickel-Cadmium battery and 5 MA on the LED, the battery would last 20 hours.)

One thing you must remember to do is check your Suunto to make sure you have not messed up the calibration of the readings. Make a rest for sighting, making sure the Suunto is unable to move. First, take a reading without adding anything to the Suunto. Second, add your LED system and repeat the above reading. The second reading should be the same as the first or you have some steel too close. Aluminum, brass, copper and some stainless steels are non-magnetic. These materials, as well as plastics can be used on the Suunto without effect. (Since one's helmet and caving lamp will presumably be left on while using the LED system, it is important to check these for effect on the instrument as well-- Ed.)

I have been using this system for over two years now in actual surveying along with the standard Suunto as the forward or backsight without any problems.

# Army Map-Folding Method Preserves Topos

by Frank Reid
Bloomington, Indiana

Topographic maps are expensive these days, and need protection for long life. Rolled and quarter-folded maps are difficult to handle, and wear out quickly.

Since 1967, I have folded all my 7.5-minute topo maps by the following procedure, one of several folding methods described in U.S. Army Field Manual FM 21-26: ***Map Reading.***

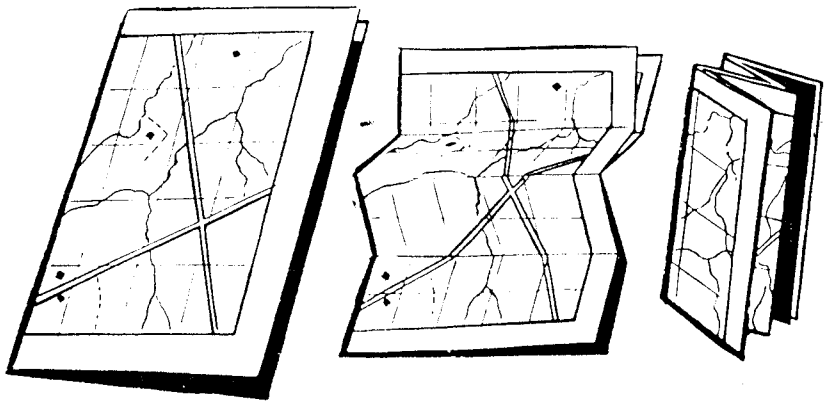
Maps folded this way are easy to store and transport, easy to sort, easy to handle in the field or car, and they last much longer because ***any spot on the map can be seen by unfolding only one crease.***

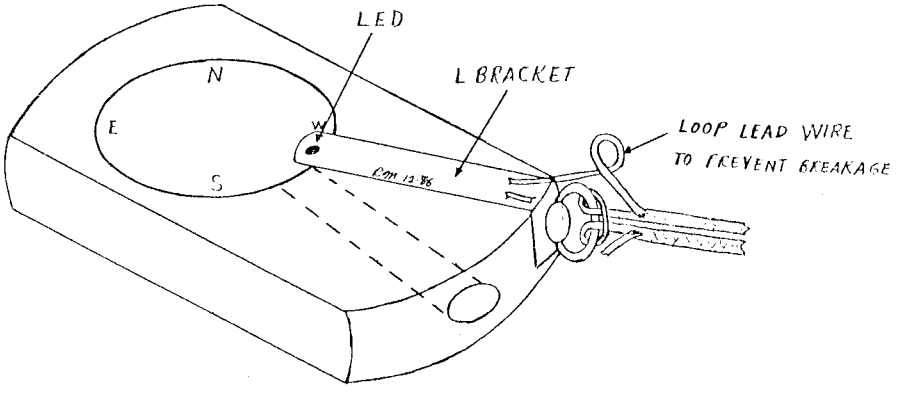
Practice first with plain paper.

1.) Fold map in half, top to bottom, with the printed side ***outward.***

2.) Facing the top portion of the half-folded map (with North upward), fold it again from side to side (vertical crease) so that the title bock in the upper right-hand corner is ***inside.***

3.) Fold each half of the quarter-folded map outward (vertical creases again), such that the upper-right title block is now on the outside (see third illustration).





**Survey & Cartography Section of the National Speleological Society**

John Ganter
Department of Geography
302 Walker Building
The Pennsylvania State University
University Park, PA 16802

Non-Profit Org.
U.S. Postage
PAID
State College, Pa.
Permit No. 138

Forwarding & Return Postage Guaranteed.
Address Correction Requested.

# SPELEO PRESS Offers New Services

Terry Raines reports that The Speleo Press has recently acquired a Linotype Series 100 composition and laser typesetting system. Built around an Apple MacIntosh Plus, the system allows word processing with MacWrite, and graphics production with MacPaint and MacDraw. The page description language is Postscript RIP.Output is produced on a Linotronic 300, with a resolution of 2540 lines per inch. Terry encourages those interested in typesetting and/or printing of books, periodicals or other cave publications to contact him. The Speleo Press, PO Box 7037, Austin, TX 78712 Phone: 512-847-2709

# UIS Commission on Karst Survey & Mapping

The Commission was reorganized during the 9th ICS, and is now chaired by Ing. Marcel Lalkovic (c/o Slovenska Speleologicka Spolocnost, Skolska 4, Liptovsky Mikulas, CSSR). In November or December 1987, the Commission will hold a Symposium entitled "Karstological Mapping in Environmental Research". Major topics will include:

* detailed karst mapping (legends, symbols, content)
* techniques of surface and underground mapping
* typology of karst regions
* mapping for economic planning and protection

The exact dates of the Symposium, to be held in Slovakia, will be announced in the next UIS Bulletin.

(From the UIS-BULLETIN, 1986-2 (30), March 1987)

# Mexico Maps For Sale

Hard-to-get topographic, geologic and general maps of Mexico are now available from Peter Sprouse. Contact him for an index map and price list: PO Box 8424, Austin, TX 78713. Phone: 512-453-4672.

# New COMPASS & TAPE Editor Sought

Cavers interested in editing COMPASS & TAPE are invited to submit a letter of application to the Section chair (John Ganter, 302 Walker Bldg., Dept. of Geography, Penn State University, University Park, PA 16802 814-865-6421). The candidate should have knowledge in the field, substantial experience in newsletter production, and access to facilities for text processing, printing, etc.